Charge-conserving denoiser

One way to impose exact charge conservation in a convolution is to ensure that all the kernel elements add to 1. Let K_I be the kernel elements where I is a 2d index tuple. If q_I is the original charge density in pixel I after convolution we have

$$q'_J = \sum_I q_I K_{J-I}.$$

Charge conservation is ensured because

$$\sum_J q'_J = \sum_J \sum_I q_I K_{J-I} = \sum_I \left(\sum_J K_{J-I} \right) q_I = \sum_I q_I.$$

And we can view the kernel K as redistributing the charge within its size.

This is of course just a linear transformation so the "best" kernel would be just some kind of averaging kernel. So next we create a combination of different kernels with space dependent weights $c_I^{(n)}$. Then we have

$$q'_J = \sum_n \sum_I c_I^{(n)} K_{J-I}^{(n)} q_I.$$

Again, we want charge conservation so

$$\sum_J q'_J = \sum_n \sum_J \sum_I c_I^{(n)} q_I K_{J-I}^{(n)} = \sum_I \left(\sum_N c_I^{(n)} \left(\sum_J K_{J-I}^{(n)} \right) q_I \right) = \sum_I \left(\sum_N c_I^{(n)} \right) q_I,$$

and if we impose

$$\sum_N c_I^{(n)} = 1$$

for all I, we get

$$\sum_J q'_J = \sum_I q_I.$$

We may also want to impose $c_I^{(n)} \geq 0$ and then we can view these coefficients as partitioning the charge in a space-dependent way. Each fraction $c_I^{(n)}$ of the charge is convolved with a different kernel $K^{(n)}$. For example, outside the streamer or in relatively flat areas we want to select a flat kernel that averages as much as possible. In regions with high gradients we prefer smaller kernels that do not smear out the charge density.

The key is that we are now almost entirely free to let the network learn to compute the partition $c_I^{(n)}$. One way to ensure that it satisfies our requirements is to first obtain other values $b_I^{(n)}$ and then pass them through a softmax:

$$c_I^{(n)} = \frac{e^{b_I^{(n)}}}{\sum_n e^{b_I^{(n)}}}.$$

The problem is then reduced to creating a network that fits together the $b_I^{(n)}$ and the kernel weights $K_I^{(n)}$.

Additional notes

1. When the charge density comes from an axi-symmetric simulation the conservation should apply to $r_I q_I$ rather than q_I alone. Here r_I is the r-coordinate of the center of the cell. We can just use that for training and inference.
2. For stability reasons and to keep a clear interpretation of the kernel weights $K_I^{(n)}$ we may also constrain them to be nonnegative.
3. If the kernels $K^{(n)}$ are not symmetric charge is not conserved even if we set a "symmetric" padding around the domain. To ensure charge conservation we can do this: (a) add a zero pad around the boundaries and (b) moving back the charge that exits the domain by reflecting it around the axis. In the code this is implemented in the `Fold2D` class, which uses matrix multiplication (perhaps not very efficient).